SANDIA REPORT

SAND2008-8057
Unlimited Release
Printed December 2008

HOPSPACK: Hybrid Optimization Parallel Search Package

Genetha A. Gray, Joshua D. Griffin, Matt Taddy, Monica Martinez-Canales, and Tamara G. Kolda

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/ordering.htm

SAND2008-8057XX
Unlimited Release
Printed December 2008

HOPSPACK: Hybrid Optimization Parallel Search Package

Genetha A. Gray
Advanced Software R & D, Dept. 8964
Sandia National Laboratories
P.O. Box 969, MS9159 Livermore, CA 94551-0969
gagray@sandia.gov

Joshua D. Griffin
Department of Numerical Optimization R&D
SAS Institute Inc.
100 SAS Campus Drive Cary, NC 27513
Joshua.Griffin@sas.com

Matt Taddy
Department of Econometrics and Statistics
The University of Chicago Graduate School of Business
5807 S. Woodlawn Ave. Chicago, IL 60637
matt.taddy@chicagogsb.edu

Monica Martinez-Canales
Department of Platform Validation Architecture
Intel Corporation
2200 Mission College Blvd. Santa Clara, CA 95054-1549
monica.martinez-canales@intel.com

Tamara G. Kolda
Informatics & Decision Sciences, Dept. 8962
Sandia National Laboratories
P.O. Box 969, MS9159 Livermore, CA 94551-0969
tgkolda@sandia.gov

Abstract

In this paper, we describe the technical details of HOPSPACK (Hybrid Optimization Parallel Search Package), a new software platform which facilitates combining multiple optimization routines into a single, tightly-coupled, hybrid algorithm that supports parallel function evaluations. The framework is designed such that existing optimization source code can be easily incorporated with minimal code modification. By maintaining the integrity of each individual solver, the strengths and code sophistication of the original optimization package are retained and exploited.

Acknowledgments

The authors would like to thank and acknowledge the following people for their contributions to this project:

- Patty Hough for her unwaivering support of these ideas throughout the lifetime of this project and for her work adapting the ideas of HOPSPACK to DAKOTA.
- Katie Fowler (Clarkson University) for her willingness to put HOPSPACK through its paces and push its development to the limit by suggesting new and innovative applications for hybrid schemes.
- Joe Castro and his LDRD team for the brainstorming and work completed in the Multifidelity Optimization (MFO) project. This work was a precursor to HOPSPACK.
- Bobby Gramacy (Cambridge) for his ideas on how to combine APPSPACK and TGP. He played a significant role in the proof of concept of the software.
- Herbie Lee (UCSC) for his contributions and ideas regarding TGP.
- American Institute of Mathematics (AIM) for their support and belief of the mathematical usefulness of this work. The week of study and research about the applicability of hybrids to hydrology was most productive.
- Cheryl Lam and Brian Owens for their financial support to continue the development of HOPSPACK.
- Chuck Hembree for his assistance in demonstrating the applicability of HOPSPACK to a set of problems related to electrical device models.

Contents

List of Figures

Chapter 1

Motivation

We are interested in solving the general nonlinear optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega \end{array} \tag{1.1}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\Omega \subset \mathbb{R}^n$ denotes set of feasible points. It is often the characteristics of Ω that dictate appropriate methods of solution. For generality, in this document, we set no restrictions on Ω and allow it to be described by bound, linear, and/or nonlinear constraints.

Picking a suitable optimization solver for 1.1 is quite challenging and has been the subject of many studies and much debate. This is because each solver comes with inherent strengths and weaknesses. For example, one may be global with slow local convergence properties. Another may have fast local convergence but is unable to perform global searches of the feasible region. In order to take advantage of the benefits of more than one solver and to try to overcome their shortcomings, two or more methods may be combined, forming a *hybrid*. Hybrid optimization is a popular approach in the combinatorial optimization community, where metaheuristics (such as genetic algorithms, tabu search, ant colony, variable neighborhood search, etc.), are combined to improve robustness and blend the distinct strengths of different approaches [1]. More recently metaheuristics have been combined with deterministic methods (such as pattern search) to form hybrids that simultaneously perform global and local searches [4, 10, 35, 45, 46, 48, 49].

Hybrid algorithms are ideal for problems in which the given feasible region consists of several subregions, each suited to a different solver. For example, solver A may be able to handle noise and undefined points while solver B may excel in smoother regions with smaller amounts of noise. Hence solver A by itself may be capable but unacceptably slow at finding a solution, whereas solver B may be unable to determine a solution given an arbitrary starting point. However, as a hybrid, solver A can be used to navigate the regions of high noise and reach a less complicate region where solver B can then be applied for fast convergence.

The HOPSPACK (Hybrid Optimization Parallel Search Package) software platform was created to facilitate the hybridization of multiple optimization routines into a single algorithm that supports parallel function evaluations. The framework was designed such that existing optimization source code may easily be incorporated with minimal modification. This approach allowed us to overcome the difficulties inherent in intrusive integration of multiple software packages including proprietary or unavailable source code and degraded quality of the original code due to extensive editing. Moreover, the HOPSPACK framework allows for optimization routines to be run simultaneously instead serially to facilitate the sharing of intermediate objective and constraint information between solvers. Moreover, this approach can exploit load balance problems of individual solvers to reduce overall latency time.

The HOPSPACK software platform allows for the hybridization of both serial and parallel methods. This allows the combination of solvers without regard for the computationally effort required to generate new iterates. For example, given two solvers where one is slow to generate new trial points and the other is fast, the slower solver is prevented from bottle-necking the optimization process by moving the point generation process to a separate processor.

This paper describes the basics of HOPSPACK. Chapter 2 reviews the creative history and development of HOPSPACK from its roots to its current state. Chapter 3 gives an overview of a classification scheme for hybrid methods so that the reader can understand where this work fits into the larger body of literature on this topic. In Chapter 4, summaries of the methods currently implemented in HOPSPACK are given. Chapter 5 reviews the software paradigm of HOPSPACK. Finally in Chapter 7, we discuss the future directions of the software.

Chapter 2

HOPSPACK: A History

Before delving into the details of the HOPSPACK software, it is helpful to first understand the foundation of the research in the area of hybrid optimization. Therefore, we include those details here and encourage the reader to peruse the referenced papers and reports for additional information.

The basis of HOPSPACK is the combination of a parallel direct search method and a second sampling scheme. This idea was first tested using Asynchronous Parallel Pattern Search (APPS), a software described in detail in Chapter 4 and in [22]. This approach has its roots in an LDRD project carried out in FY04 and FY05 by PI Joe Castro called MFO. As part of this project, APPS was placed in an an oracle framework to achieve the goals of combining multiple models. In optimization methods, oracles are used to predict points at which a decrease in the objective function might be observed. Analytically, an oracle is free to choose points by any finite process. (See [25] and references therein.) Note these points are given in addition to those generated by the basic pattern search and therefore,convergence is not adversely affected. The details of this project, the software implementation of the oracle within APPSPACK and some results can be found in [3] and in [19].

The positive results observed in the MFO project prompted some new ideas in FY06 as part of the PRIDE LDRD lead by Monica Martinez-Canales. The goal of this project was to make the Sandia simulation-based design process tractable and reduce the computational costs associated with design optimization. This project built on the APPS-oracle software design by introducing the concept of a Gaussian Process as an oracle. In conjunction with statisticians collaborators, a proof-of-concept code was implemented and tested. These results are presented in [17] and in the project final report [33]. Note that this aspect of the pride project was a first step towards creating an appropriate optimization under uncertainty algorithm which would allow the user to perform both calibration and uncertainty quantification at the same time.

In FY07, HOPSPACK hit its stride. Thanks to funds from two different ASC projects (ASC-V&V and ASC-Outer Loop), a more robust software implementation of the APPS-oracle design was started. It was during these two years that HOPSPACK grew to its current state. Moreover, we were able to demonstrate its usefulness for a Sandia application as shown in [20, 43]. These successes prompted the DAKOTA development team to investigate the inclusion of this approach to hybridization in the DAKOTA toolkit. These successes prompted renewed support from ASC in FY08 as well as some support from an ASCR project focused on derivative-free optimization.

Chapter 3

Classifying Hybrid Methods

The HOPSPACK software is an example of a hybrid optimization scheme. To better understand its functions, we put it in the context of a general classification scheme. Raidl [36, 37] outlines a method for categorizing hybrid optimization methods using four main characteristics: 1) class of algorithms used to form hybrids, 2) level of hybridization, 3) order of execution, and 4) control strategy. Although this classification scheme was devised to describe metaheuristics used by the combinatorial optimization community, it is also applicable here and thus we use it to describe HOPSPACK.

3.1. Class of algorithms

Currently, HOPSPACK includes four algorithms: a generating set search (GSS), a deterministic sampling, Latin Hypercube sampling, and a Gaussian process. Each of these methods are described in detail in Chapter 4.

HOPSPACK originated in a project whose aim was to solve an optimization problem in which computing the objective function value relied on the output of a numerical simulator. Derivatives were unavailable and approximate derivatives were unreliable. A variety of derivative-free optimization methods have emerged and matured over the years to address such problems, but for this problem, none could both sufficiently sample the design space while limiting the computational burden. Therefore, we chose to combine two methods– one computationally inexpensive and one with robust sampling. This idea is the basis of HOPSPACK and defines the current procedure in which the methods are running simultaneously and the GSS is constantly updating itself with the best point found by any method. Future plans for HOPSPACK include generalizing this framework and are discussed in more detail in Section 6.

3.2. Level of Hybridization

Hybrid algorithms can classified as loosely or tightly coupled. In general, loosely coupled approaches retain the individual identities of the methods being hybridized. In contrast, tightly coupled hybrids exhibit a strong relationship between the individual pieces and may share components or functions [37]. The HOPSPACK platform facilitates loosely coupled methods in that the individual components essentially run independently of one another. Components are in fact interchangeable.

Loosely coupled hybrids are advantageous from both a software development and theoretical perspective. Because significant work is required to develop a single state-of-the-art optimization routine, it would be impractical and bug prone to attempt to re-implement existing methods as part of a tightly coupled algorithm. Instead, each method used in the hybrids described in this paper use the original source code with minimal changes. This approach also keeps theoretical convergence properties intact.

3.3. Order of Execution

The order of execution of hybrid algorithms can either be sequentially or parallel. Sequentially hybrid methods string together a set of algorithms head to tail, using the results of a completed run of one algorithm to seed the next. From this perspective it is often unclear whether or not the previously executed algorithms should be viewed simply as a preprocessing step, or if ensuing algorithm runs should be viewed as post-processing. On the other hand, parallel hybrids execute the individual methods simultaneously and can thus be made to be collaborative and share information dynamically to improve performance. From this perspective, HOPSPACK is primarily a parallel hybrid. However, the platform is set-up so that LHS can be used as a preprocessing step prior to the full-fledge optimization. It is of course possible to continue to use the results of subsequent LHS runs throughout the optimization process; something that may be beneficial if a large number of evaluation processors is available.

3.4. Control Strategy

The control strategy of hybrid algorithms can be either integrative or collaborative. In an purely integrative approach, one individual algorithm is subordinate to or an embedded component of another. Collaborative methods give equal importance and control to both algorithms. Algorithms merely exchange information instead of being an integral part of one another. One of the most common algorithms that uses a collaborative control strategy combines highly similar (or homogeneous) algorithms to optimize different parts of the feasible region in parallel. The hybrid approaches available in HOPSPACK are primarily collaborative.

Chapter 4

HOPSPACK Algorithms

HOPSPACK currently includes four methods for generating new iterates. In this Chapter, we describe each in detail.

4.1. Generating Set Search (APPSPACK)

Generating set search (GSS) denotes a class of algorithms for bound and linearly constrained optimization that obtain conforming search directions from generators of local tangent cones [28, 32]. A GSS falls into the category of direct search optimization methods [27] in which algorithms are derivative-free and make no attempt to explicitly evaluate, estimate, or model local derivatives. Derivative-free approaches tend to be more robust for problems that are noisy, nonsmooth, discontinuous, and/or contain undefined “feasible” points than their derivative-based counterparts which often break down at points where derivatives cease to exist. It is important to note that this feature is of direct search methods is both a strength and weakness. Without derivatives, such algorithms are only practical for a small (e.g. typically less than a hundred) number of parameters. However, many direct search algorithms, including GSS, offer rigorous convergence properties when derivatives do in fact exists.

In the case that only bound constraints are present, GSS is identical to a pattern search optimization algorithm. Pattern searches use a predetermined pattern of points to sample a given function domain. Since the majority of the computational cost of pattern search methods is the function evaluations, parallel pattern search (PPS) techniques have been developed to reduce the overall computation time. Specifically, PPS exploits the fact that once the points in the search pattern have been defined, the function values at these points can be computed simultaneously [8, 44]. The APPS (Asynchronous Parallel Pattern Search) algorithm and corresponding APPSPACK software are one variant of PPS specifically developed to address load balance issues in a parallel computing environment [21, 26, 29]. The APPS optimization algorithm has been shown to have identical convergence properties to synchronous generating set search and can significantly reduce run time [22].

Implementation of the APPS algorithm is more complicated than a basic GSS in that it requires careful bookkeeping. However, the details are irrelevant to the overall understanding of the hybrid scheme that is the focus of this paper and thus we present a basic GSS in Algorithm 1. Interested reader are directed to [21, 26] for a detailed description and analysis of the APPS algorithm and corresponding APPSPACK software.

Algorithm 1 A basic GSS algorithm

```
 1: Let $x_0$ be the *starting point*.
 2: Let $\Delta_0$ be the *initial step size*.
 3: Let $\mathcal{D}$ be the set of *positive spanning directions*.
 4: **while** Not converged **do**
 5:     Generate trial points $Q_k = \{x_i + \widetilde{\Delta}_k d_i \,\big|\, 1 \leq i \leq |\mathcal{D}|\}$
 6:     where $\widetilde{\Delta}_k \in [0, \Delta_k]$ denotes maximum feasible step along $d_i$.
 7:     Evaluate trial points (possibly in parallel)
 8:     **if** $\exists\, x_q \in Q_k$ such that $f(x_q) - f(x_k) < \alpha\Delta_k^2$ **then**
 9:         $x_{k+1} = x_q$                              ▷ Iteration was successful
10:     **else**
11:         $x_{k+1} = x_k$                              ▷ Iteration is unsuccessful
12:         $\Delta_{k+1} = \Delta_k/2$                  ▷ Reduce step size
13:     **end if**
14: **end while**
```

Note that in a basic GSS, after a successful iteration (one in which a new best point has been found), the step size is either left unchanged or increased. In contrast, when the iteration was unsuccessful, the step-size is necessarily reduced. A defining difference between the basic GSS and APPS is that the APPS algorithm processes the directions independently, and each direction may have its own corresponding step-size. Global convergence to locally optimal points is ensured using a *sufficient decrease criteria* for accepting new best points. A trial point $x_k + \Delta d_i$ is considered better than the current best x_k point if

$$f(x_k + \Delta d_i) - f(x_k) < \alpha\Delta^2,$$

for $\alpha > 0$. Because APPS processes search direction independently, it is possible that the current best point is updated to a new better point before all the function evaluations associated with a set of trial points Q_k have been completed. These results are referred to as *orphaned points* as they are no longer tied to the current search pattern and attention must be paid to ensure that the sufficient decrease criteria is applied appropriately. The support of these orphan points is a feature of the APPS algorithm which makes it naturally amenable to a hybrid optimization structure. Iterates generated by alternative algorithms can be simply be treated as orphans without the loss of favorable theoretical properties or local convergence theory of APPS. It is important to note that this paradigm is in fact extensible to many other optimization routines.

4.2. Dividing Rectangles (DIRECT)

DIRECT, an acronym for DIviding RECTangles, is a deterministic sampling algorithm developed in [24] that performs global optimization on bound constrained problems. Since its introduction in the early 1990's, a significant number of papers have been written analyzing, describing, and developing new variations of this highly effective algorithm. Some of these include [2, 5, 6, 11–15, 23, 30, 41, 46, 47].

The basic DIRECT algorithm is outlined in Algorithm 2. Note that it requires that both the upper and lower bounds are finite. As its name suggests, the DIRECT algorithm is a partitioning algorithm that sequentially refines a rectangular partition of the feasible region at each iteration selecting an *optimal* hyper-rectangle to trisect. The algorithm begins by mapping the rectangular feasible region onto the the unit hypercube; that is DIRECT optimizes the transformed problem

$$\begin{aligned} \min_{\widetilde{x}\in\mathbb{R}^n} \quad & \widetilde{f}(\widetilde{x}) = f(S\widetilde{x}+\ell) \\ \text{subject to} \quad & 0 \le \widetilde{x} \le e, \end{aligned} \tag{4.1}$$

where $\widetilde{x} = S^{-1}(x-\ell)$ with $S = \mathrm{diag}(u_1-\ell_1,\ldots,u_n-\ell_n)$. Figure 1 illustrates four iterations of DIRECT for a two dimensional example. At each iteration, a candidate, potentially optimal, hyper-rectangle is selected and refined. Though other stopping criteria exist, this process typically continues until a user defined budget of function evaluations has been expended.

```
1: while not converged do
2:     Let $\mathcal{S}_k$ denote the set of potentially optimal hyper-rectangles.
3:     Select any hyper-rectangle $R$ from $\mathcal{S}_k$; let $c$ denote its center.
4:     Generate trial points
           $Q_k = \left\{ c \pm \frac{1}{3}\Delta_k e_i \,\middle|\, i \in \mathcal{M} \right\}$
       Here $\mathcal{M}$ denotes the set of indices corresponding to sides of $R$
       with maximum length $\Delta_k$.
5:     Evaluate trial points.
6:     Define $w_i = \min(\widetilde{f}(c+\frac{1}{3}\Delta_k e_i), \widetilde{f}(c-\frac{1}{3}\Delta_k e_i))$.
7:     Beginning with smallest $w_i$ and working towards the largest $w_i$,
       trisect $R$ along dimension $i$.
8: end while
```

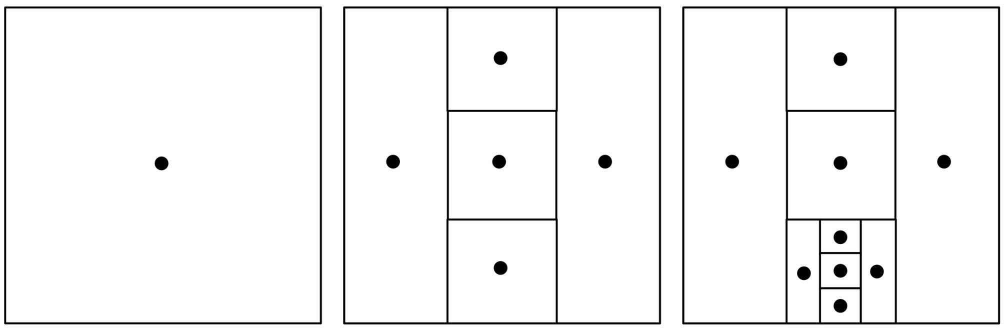

Figure 4.1. DIRECT iteratively subdivides the optimal hyper-rectangle into thirds.

Definition 1 describes the criteria for being a potentially optimal hyper-rectangle given a constant $\varepsilon > 0$.

Definition 1 (Potentially optimal hyper-rectangle [24]) *Suppose there are K enumerated hyper-rectangles subdividing the unit hypercube from* **??***eq:nlptransformed](***??***eq:nlptransformed) with centers c_i, $1 \leq i \leq K$. Let γ_i denote the corresponding distance from the center c_i to its vertices. A hyper-rectangle ℓ is considered potentially optimal if there exists $\alpha_K > 0$ such that*

$$\begin{aligned} \widetilde{f}(c_\ell) - \alpha_K \gamma_\ell &\leq \widetilde{f}(c_i) - \alpha_K \gamma_i, \; 1 \leq i \leq K && (4.2) \\ \widetilde{f}(c_\ell) - \alpha_K \gamma_\ell &\leq \widetilde{f}_{\min} - \varepsilon |f_{\min}|. && (4.3) \end{aligned}$$

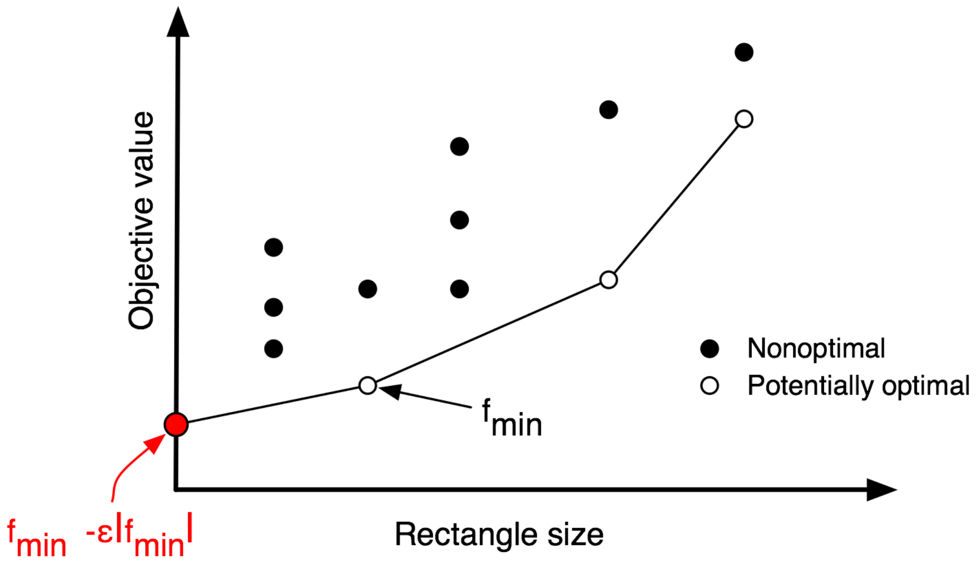

Figure 4.2. Potentially optimal hyper-rectangles can be found by forming the convex hull of the set $\{f(c_i), \gamma_i\}$, where c_i denotes the center point of the ith hyper-rectangle and γ_i the corresponding distance to hyper-rectangle's vertices.

The set of potentially optimal hyper-rectangles forms a convex hull for the set point $\{\widetilde{f}(c_i), \gamma_i\}$. **??**fig:dirhull]Figure **??**fig:dirhull illustrates this. Notice that the user defined parameter ε controls whether or not the algorithm performs more of a global or local search.

4.3. Treed Gaussian Process (TGP)

Statistical emulation can serve as an alternative to traditional optimization methods. To accomplish this, the function values associated with previously evaluated iterates are used to train a statistical model, and this model is used to draw inferences about the location of an optimum. The idea of using a stochastic process to approximate an unknown function dates back as far as Poincaré in the 19th century [9]. In particular, a Gaussian Process (GP) is typically used for the emulation of computer simulators [38, 39]. The output of the simulator is treated as a random variable $Y(\mathbf{x})$ that depends on the input vector **x** such that the response varies smoothly. This smoothness is given by the covariance structure of the GP. The mean and covariance functions determine the characteristics of the process, as any finite set of locations has a joint multivariate Gaussian distribution. For more details about GPs, interested readers are directed to references such as [7] and [42]. A Bayesian approach to GPs allows full estimation of uncertainty, which is useful when trying to determine the probability that an unsampled location will be an improvement over the current known optimum.

Standard GP models have several drawbacks, including strong assumptions of stationarity and poor computational scaling. To reduce these problems, treed Gaussian process (TGP) models partition the domain using a recursive tree structure and then fit independent GP models within each partition. An implementation of this idea is described in [16] and available in the form of a `tgp` library for the open source statistical package R
(see `http://www.cran.r-project.org/src/contrib/Descriptions/tgp.html`).

Within HOPSPACK, TGP can be used to build a globally oriented pattern of input locations. The selection of candidates to be included in the hybrid scheme is based on predicted improvement statistics at candidate **x** input locations, $I^g(\mathbf{x}) = \max\{(f_{best} - f(\mathbf{x}))^g, 0\}$, where f_{best} is the response corresponding to the current best point, and g is a positive integer. In particular, a predetermined number m of locations can be chosen for evaluation through maximization (over a discrete candidate set) of the expected multi-location improvement, $\mathbb{E}[I^g(x_1, \ldots, x_m)]$, where

$$I^g(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \max\{(f_{best} - f(x_1))^g, \ldots, (f_{best} - f(\mathbf{x}_m))^g, 0\}$$

[40].

4.4. Latin Hypercube Sampling (LHS)

A space filling sample of the bounded input region is required for the statistical generation of search patterns in the previous section. In addition, it can be valuable to search the input space before optimization begins in order to choose starting locations and build a decent training set for the statistical model. The literature on this subject is vast (see for example [39] and references therein) and specific application could depend on the modeling approach. We have used Latin hypercube sampling (see for example [34]) as an efficient way to sample the input space. This is straightforward to implement over any bounded region, and a design can be formed by taking independent LHS in each input dimension and using the combined samples as a set of input vectors to be evaluated.

Chapter 5

Software Paradigm

HOPSPACK supports hybrid optimization using a mediator/citizen/conveyor paradigm. Citizens are used to denote arbitrary optimization tools or solvers: in this case, LHS, DIRECT, TGP, and APPS. They communicate with a single mediator object, asynchronously exchanging unevaluated trial points for completed function evaluations. A Citizen may choose to generate trial-point in-line or on separate citizen-worker processors altogether.

The mediator ensures that points are evaluated in a predefined order specified by the users and iteratively exchanges an ordered queue of unevaluated trial points for newly completed function evaluations with a conveyor object. This mediator has no knowledge of the underlying optimization problem and is only concerned with managing function evaluations. Three basic steps are performed iteratively: 1) exchange evaluated points for unevaluated points with citizens, 2) prioritize the list of requested evaluations, and 3) exchange unevaluated points for evaluated points with conveyor. This process continues until either a maximum budget of evaluations has been reached or all citizens have indicated they are finished. Along with a list of supported citizens, the user may provide a corresponding priority list that the mediator will use when ordering the evaluation queue. Citizens with higher priority will have their points moved towards the front of the evaluations queue, while citizens with the same priority will be spliced.

The purpose of the conveyor is to evaluation points from a given queue of points provided by the mediator asynchronously in parallel in FIFO. The conveyor seeks to maximize the efficient use of available evaluation processors and to reduce as best as possible processor idle time. Thus while evaluation processors run idle, the conveyor will submit points from the queue to be evaluated. The conveyor also stores a function value cache that lexicographically stores a history of all completed function evaluations using a splay tree as described in [21]; this prevents a linear increase in look up time. Thus, prior to submitting a point to be evaluated, the cache is queried to determine if the given point has already previously been evaluated. If the point is not currently cached, a second query is performed to determine if an equivalent point is currently being evaluated; if so, the point is added to a pending list and awaits completion of the corresponding evaluation. The conveyor thus guarantees that given a set of equivalent points, only one of these points will actually be evaluated, while the rest are all assigned this returned value, regardless of submission time.

Using this paradigm, for the approach outlined in this paper, we use three citizens: LHS, DIRECT, and APPS. The LHS citizen is given highest priority and hence has all of the points (of its

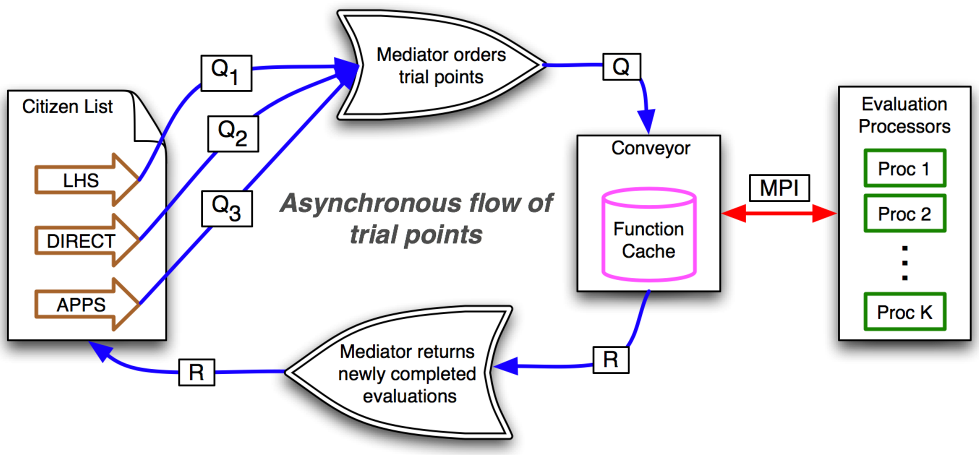
Citizen List
LHS
DIRECT
APPS
Q1
Q2
Q3
Mediator orders trial points
Q
Conveyor
Function Cache
MPI
Evaluation Processors
Proc 1
Proc 2
Proc K
Asynchronous flow of trial points
R
Mediator returns newly completed evaluations
R

Figure 5.1. Points are submitted and evaluated asynchronously: no method can force another to wait. Evaluation processors never run idle unless all citizens have submitted empty trial point queues.

one time submission) are evaluated first. However, without load imbalance, the moment processors become available that are no longer needed by LHS, DIRECT and APGSS begin the optimization process. DIRECT and APGSS are given equal priority and hence points are spliced so that the trial-point evaluation rate of each is roughly the same. An alternate ordering strategy would be to give DIRECT higher priority in the evaluation queue early in the optimization process, and gradually increase the importance of the APPS, the local solver. This could be based upon the number of completed evaluations and the maximum allowed for a given solve.

An immediate benefit of this approach is load balance. While DIRECT can be ran in parallel, it does suffer from load imbalance in that trial-points are submitted in batches. The next iteration of DIRECT cannot continue until all points in the previous batch have completed. Thus, suppose for example, that DIRECT submits 9 trial-points and 8 evaluations processors are available and that each function evaluation takes 1 hour. Then after an hours time, 8 of these evaluations will be complete. But DIRECT cannot continue until all 9 points in the current batch are completed; the final point will thus take another hours time, letting 7 processors run idle the entire time. Thing become even worse and more unpredictable when evaluation time is dramatically different, i.e. points periodically crashing the simulation and returning immediately.

In contrast, using the current mediator/citizen/conveyor paradigm, in the previous example, while direct is using only one processors, APGSS has all 7 processors at its disposal. Further, because APGSS is asynchronous, any algorithm paired with APGSS in this manner creates a new asynchronous hybrid. Extra solvers paired with APGSS may be seen simply as alternate means for generating periodically generating new trial points.

Figure 5 illustrates the flow of trial points from the citizens, through the mediator, to the conveyor, and back to the citizens. The stream of trial-points is asynchronous in the sense that citizens can always submit points at each iteration, they cannot however, force another citizen to wait if evaluation processors are available. Here Q_1, Q_2, and Q_3 denote trial points submitted by citizens LHS, DIRECT, and APPS respectively, while Q is used to denote the ordered list of trial points given to the conveyor. R stores recently completed evaluations. Citizens are permitted to view all completed function evaluations.

Chapter 6

The Future of HOPSPACK

We have shown HOPSPACK to be quite successful on both test problems and real applications (see [20, 31, 43]). Unfortunately, funding shortfalls have prevented us from preparing and releasing the HOPSPACK software as an open source package for general use. However, research continues in this area. The software ideas are being implemented for use as part of Sandia's DAKOTA and ACRO toolkits. The algorithms are being applied to groundwater management problems [18]. New funding sources are being sought in the areas of statistics, optimization, sampling under uncertainty, risk analysis, environmental clean-up and energy transportation. Interested readers should stay tuned for future results.

References

[1] Enrique Alba, editor. *Parallel Metaheuristics*. John Wiley & Sons, Inc., 2005.

[2] M. C. Bartholomew-Biggs, S. C. Parkhurst, and S. R. Wilson. Global optimization - stochastic or deterministic? In K Albrecht, A; Steinhofel, editor, *SAGA 2003: 2nd Intl. Symposium on Stochastic Algorithms*, volume 2827 of *Lecture Notes in Computer Science*, pages 125 – 137. Springer-Verlag, 2003.

[3] J. Castro, G. A. Gray, A. Guinta, and P. Hough. Developing a computationally efficient dynamic multilevel hybrid optimization scheme using multifidelity model interactions. Technical Report SAND2005-7498, Sandia National Labs, Nov. 2005.

[4] Joseph P. Castro, Genetha A. Gray, Anthony A. Giunta, and Patricia D. Hough. Developing a computationally efficient dynamic multilevel hybrid optimization scheme using multifidelity model interactions. Technical Report SAND2005-7498, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, November 2005.

[5] Lakhdar Chiter. Direct algorithm: A new definition of potentially optimal hyperrectangles. *Applied Mathematics and Computation*, 179(2):742–749, 2006.

[6] Steven E. Cox, William E. Hart, Raphael Haftka, and Layne Watson. DIRECT algorithm with box penetration for improved local convergence. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002. AIAA-2002-5581.

[7] N. A. C. Cressie. *Statistics for Spatial Data, revised edition*. John Wiley & Sons, 1993.

[8] J. E. Dennis, Jr. and V. Torczon. Direct search methods on parallel machines. *SIAM J. Optim.*, 1:448–474, 1991.

[9] P. Diaconis. Bayesian numerical analysis. In S.S. Gupta and J.O. Berger, editors, *Statistical Decision Theory and Related Topics IV*. Springer-Verlag, 1988.

[10] Shu-Kai S. Fan and Erwie Zahara. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research*, 181(2):527–548, 2007.

[11] D. E. Finkel and C. T. Kelley. An adaptive restart implementation of direct. Technical Report CRSC-TR04-30, Center for Research in Scientific Computation, North Carolina State University, August 2004.

[12] D. E. Finkel and C. T. Kelley. Convergence analysis of the DIRECT algorithm. Technical Report CRSC-TR04-28, CSRC, NSCU, 2004.

[13] D. E. Finkel and C. T. Kelley. Additive scaling and the DIRECT algorithm. *J. Global Optim.*, 36(4):597–608, 2006.

[14] J. M. Gablonsky. *Modifications of the DIRECT algorithm*. PhD thesis, North Carolina State University, 2001.

[15] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *J. Global Optim.*, 21(1):27–37, September 2001.

[16] Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *J. Amer. Statist. Assoc.*, 2008. to appear.

[17] G. A. Gray et al. Designing dedicated experiments to support validation and calibration activities for the qualification of weapons electronics. In *Proceedings of the 14th NECDC*, 2007. also available as Sandia National Labs Technical Report SAND2007-0553C.

[18] G. A. Gray, K. Fowler, and J. D. Griffin. Derivative-free optimization for hydrological applications, October 2008. Presentation at the INFORMS Annual Meeting in Washington D.C.; slides available as SAND2008-7883P.

[19] G. A. Gray and K. R. Fowler. Approaching the groundwater remediation problem using multifidelity optimization. In *Proc. of the CMWR XVI- Computational Methods in Water Resources*, June 2006.

[20] G. A. Gray, M. Taddy, J. D. Griffin, M. Martinez-Canales, and H. K. H. Lee. Hybrid optimization: A tool for model calibration. Technical Report SAND2008-0145J, Sandia National Labs, 2008. submitted to *SIAM J. Optim.*

[21] Genetha A. Gray and Tamara G. Kolda. Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization. *ACM T. Math. Software*, 32(3):485–507, 2006.

[22] Joshua D. Griffin, Tamara G. Kolda, and Robert Michael Lewis. Asynchronous parallel generating set search for linearly-constrained optimization. Technical Report SAND2006-4621, Sandia National Laboratories, Albuquerque, New Mexico and Livermore, California, August 2006.

[23] D R. Jones. The direct global optimization algorithm. In *Encyclopedia of Optimization*, volume 1, pages 431–440. Kluwer Academic Boston, 2001.

[24] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optimiz. Theory App.*, 79(1):157–181, 1993.

[25] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45(3):385–482, 2003.

[26] Tamara G. Kolda. Revisiting asynchronous parallel pattern search for nonlinear optimization. *SIAM J. Optimiz.*, 16(2):563–586, December 2005.

[27] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Rev.*, 45(3):385–482, August 2003.

[28] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Stationarity results for generating set search for linearly constrained optimization. *SIAM J. Optimiz.*, 17(4):943–968, 2006.

[29] Tamara G. Kolda and Virginia Torczon. On the convergence of asynchronous parallel pattern search. *SIAM J. Optimiz.*, 14(4):939–964, 2004.

[30] Chiter Lakhdar. Towards a new direct algorithm: a two-points based sampling method. Available from `http://www.optimization-online.org/DB_FILE/2005/03/1077.pdf`, 2005.

[31] H. K. H. Lee, R. Grammacy, M. Taddy, and G. A. Gray. *Applied Bayesian Analysis*, chapter Designing and Analyzing a circuit device experiment using treed Gaussian process. Oxford University Press, to appear. currently available as SAND2008-3869P.

[32] Robert Michael Lewis, Anne Shepherd, and Virginia Torczon. Implementing generating set search methods for linearly constrained minimization. Technical Report WM-CS-2005-01, Department of Computer Science, College of William & Mary, Williamsburg, VA, July 2005. Revised July 2006.

[33] M. Martinez-Canales et al. Penetrator reliability investigation and design exploration: From conventional design processes to innovative uncertainty-capturing algorithms. Technical Report RAA5248161, Sandia National Labs, Nov. 2006.

[34] M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables of output from a computer code. *Technometrics*, 22:239–245, 1979.

[35] Joshua L. Payne and Margaret J. Eppstein. A hybrid genetic algorithm with pattern search for finding heavy atoms in protein crystals. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 377–384, New York, NY, USA, 2005. ACM.

[36] Jakob Puchinger and Günther R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *IWINAC: Proceedings of First International Work-conference on the Interplay between Natural and Artificial Computation*, pages 41–53, 2005.

[37] Günther R. Raidl. A unified view on hybrid metaheuristics. In *HM06: Third International Workshop on Hybrid Metaheuristics*, pages 1–12, 2006.

[38] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statist. Sci.*, 4:409–435, 1989.

[39] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

[40] M. Schonlau, W. Welch, and D. Jones. Global versus local search in constrained optimization of computer models. In N. Flournoy, W. Rosenberger, and W. Wong, editors, *New Developments and Applications in Experimental Design*, volume 34, pages 11–25. Institute of Mathematical Statistics, 1998.

[41] E. S. Siah, M. Sasena, J. L. Volakis, P. Y. Papalambros, and R. W. Wiese. Fast parameter optimization of large-scale electromagnetic objects using DIRECT with Kriging metamodeling. *IEEE T. Microw. Theory*, 52(1):276 – 285, January 2004.

[42] Michael L. Stein. *Interpolation of Spatial Data*. Springer, New York, NY, 1999.

[43] M. Taddy, H. K. H. Lee, G. A. Gray, and J. D. Griffin. Bayesian guided pattern search for robust local optimization. Technical Report SAND2008-0104J, Sandia National Labs, 2008. submitted to *Technometrics*.

[44] V. Torczon. PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report TR92-09, Rice Univ., Dept. Comput. Appl. Math., Houston, TX, 1992.

[45] A. Ismael F. Vaz and Lus N. Vicente. A particle swarm pattern search method for bound constrained nonlinear optimization. Technical report, Department of Mathematics, University of Coimbra, 2006.

[46] K. P. Wachowiak and T. M. Peters. Combining global and local parallel optimization for medical image registration. In JM Fitzpatrick, JM; Reinhardt, editor, *Medical Imaging 2005: Image Processing*, volume 5747, pages 1189 – 1200. SPIE, April 2005.

[47] M. P. Wachowiak and T. M. Peters. Parallel optimization approaches for medical image registration. In *MICCAI 2004: 7th International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 3216 of *Lecture Notes in Computer Science*, pages 781 – 788. Springer-Verlag, 2004.

[48] Peng Yehui and Liu Zhenhai. A derivative-free algorithm for unconstrained optimization. *Applied Mathematics - A Journal of Chinese Universities*, 20(4):491–498, December 2007.

[49] T. Zhang, K. K. Choi, S. Rahman, K. Cho, P. Baker, M. Shakil, and D. Heitkamp. A hybrid surrogate and pattern search optimization method and application to microelectronics. *Structural and Multidisciplinary Optimization*, 32(4):327–345, October 2006.

DISTRIBUTION:

1	MS 9159 Genetha Gray, 8964
1	MS 9159 Patty Hough, 8964
1	MS 9159 Tammy Kolda, 8962
1	MS 9159 Heidi Ammerlahn, 8962
1	MS 9152 Mike Hardwick, 8964
1	MS 1318 Laura Swiler, 1411
1	MS 1318 Mike Eldred, 1411
1	MS 1318 Jim Stewart, 1411
1	MS 1318 Bill Hart, 1412
1	MS 0370 John Siirola, 1433
3	MS 9018 Central Technical Files, 8945-1
2	MS 0899 Technical Library, 9616
1	MS 0612 Review & Approval Desk, 9612
1	MS 0123 D. Chavez, LDRD Office, 1011